

DevOps & MLOps

Activity 1

Create an ML model for the diabetes data and deploy using Docker.

Prerequisites:

VS Code
VS Code Extension – Python
Docker
Docker Hub

Step 1 (model.py)

Creation of ML Model and Pickle(.pkl) file

Note: After the successful completion, model.pkl will be generated.

```
import pickle

from sklearn import datasets

iris=datasets.load_iris()

x=iris.data

y=iris.target

#labels for iris dataset

labels ={

    0: "setosa",

    1: "versicolor",

    2: "virginica"

}

#split the data set

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.60)


#Using decision tree algorithm

from sklearn import tree

classifier=tree.DecisionTreeClassifier()

classifier.fit(x_train,y_train)

predictions=classifier.predict(x_test)
```

```
#export the model
pickle.dump(classifier, open('model.pkl','wb'))

#load the model and test with a custom input
model = pickle.load( open('model.pkl','rb'))
x = [[6.7, 3.3, 5.7, 2.1]]
predict = model.predict(x)
print(predict)
print("Hello Worlds")
print(labels[predict[0]])
```

Step 2 (server.py)

Creation of UI and Web Framework using Streamlit

```
import streamlit as st
import pickle
model = pickle.load(open('model.pkl', 'rb'))

def predict(sl,sw,pl,pw):
    prediction=model.predict([[sl,sw,pl,pw]])
    return prediction

def main():
    st.title("IRIS Prediction")
    html_temp = """
    <div style="background-color:tomato;padding:10px">
    <h2 style="color:white;text-align:center;">Streamlit IRIS Predictor </h2>

    </div>
    """
    st.markdown(html_temp,unsafe_allow_html=True)
```

```
sl = st.text_input("Sepal Length","Type Here")
sw = st.text_input("Sepal Width","Type Here")
pl = st.text_input("Petal Length","Type Here")
pw = st.text_input("Petal Width","Type Here")
result=""

if st.button("Predict"):
    result=predict(sl,sw,pl,pw)
st.success('The output is {}'.format(result))

if st.button("About"):
    st.text("Lets LEarn")
    st.text("Built with Streamlit")
main()
```

(to run this server.py use -> streamlit run server.py (or) python -m streamlit server.py)

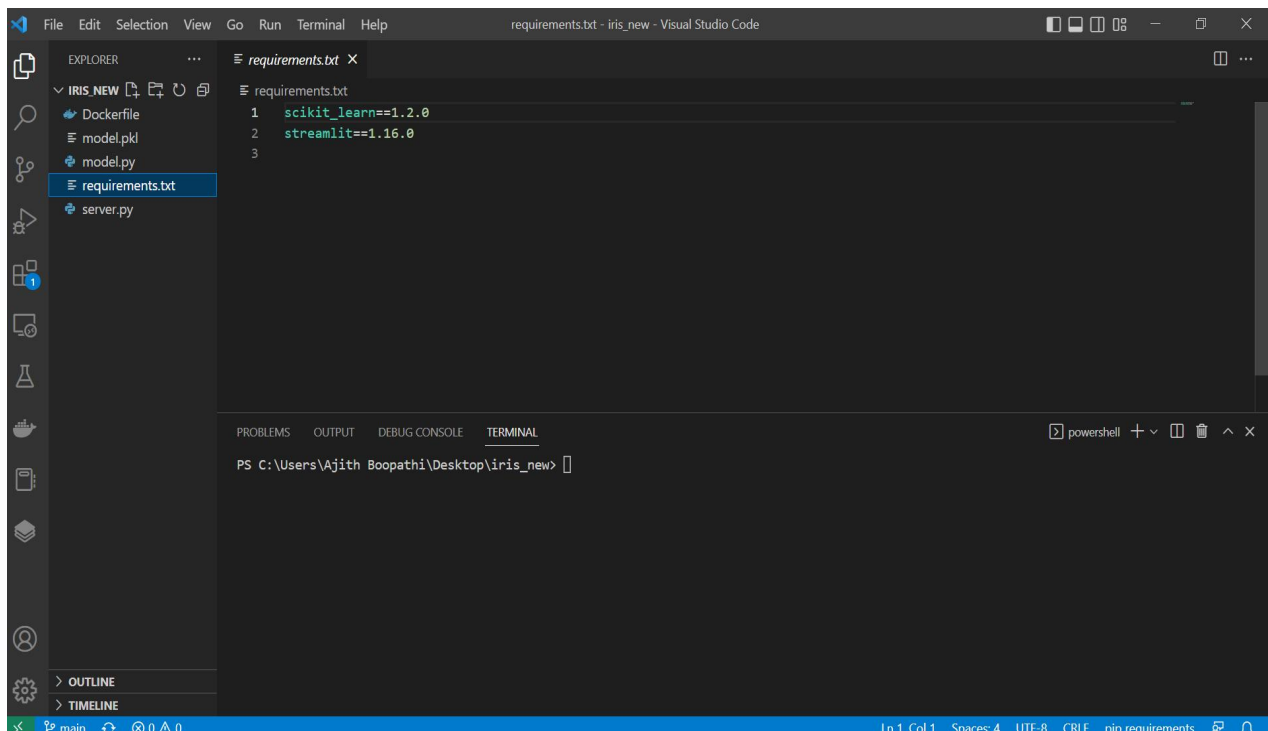
Step 3

Generate requirement file

Note

- Run the below script in VS Code terminal window of your project directory.
- After the successful completion, it generates a requirements.txt file.

- pip install pipreqs (or) python -m pip install streamlit
- pipreqs . (or) python -m pipreqs.pipreqs



The screenshot shows the Visual Studio Code interface. In the Explorer panel on the left, the file 'requirements.txt' is selected under the 'IRIS_NEW' folder. The main editor displays the contents of 'requirements.txt' with the following text:

```
requirements.txt
1  scikit_learn==1.2.0
2  streamlit==1.16.0
3
```

Below the editor, the TERMINAL panel is open, showing a PowerShell prompt at the directory 'C:\Users\Ajith Boopathi\Desktop\iris_new'.

Step 4

Creation of Docker file.

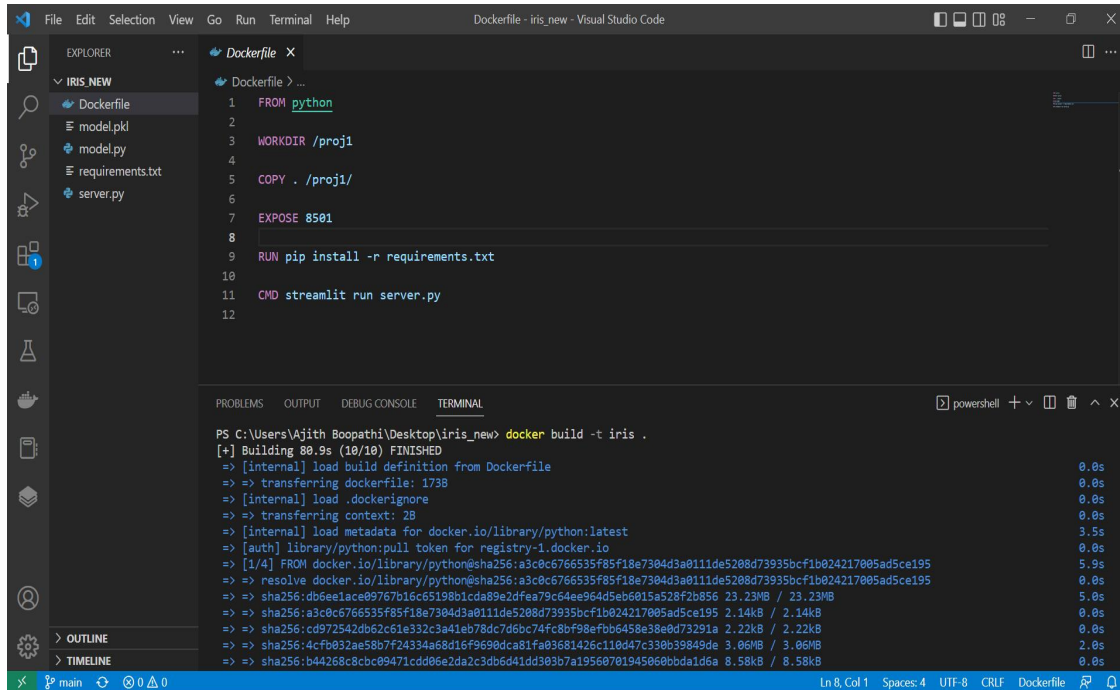
FROM python
 WORKDIR /pythondir
 COPY . /pythondir
 EXPOSE 8501
 RUN pip install -r requirements.txt
 CMD streamlit run app.py

Step 5

Run the following script in VS Code terminal

docker build -t iris .

docker run -p 8501:8501 iris



The screenshot shows the VS Code interface with a Dockerfile open in the editor and a terminal window at the bottom. The Dockerfile contains the following instructions:

```

1 FROM python
2
3 WORKDIR /proj1
4
5 COPY . /proj1/
6
7 EXPOSE 8501
8
9 RUN pip install -r requirements.txt
10
11 CMD streamlit run server.py
12

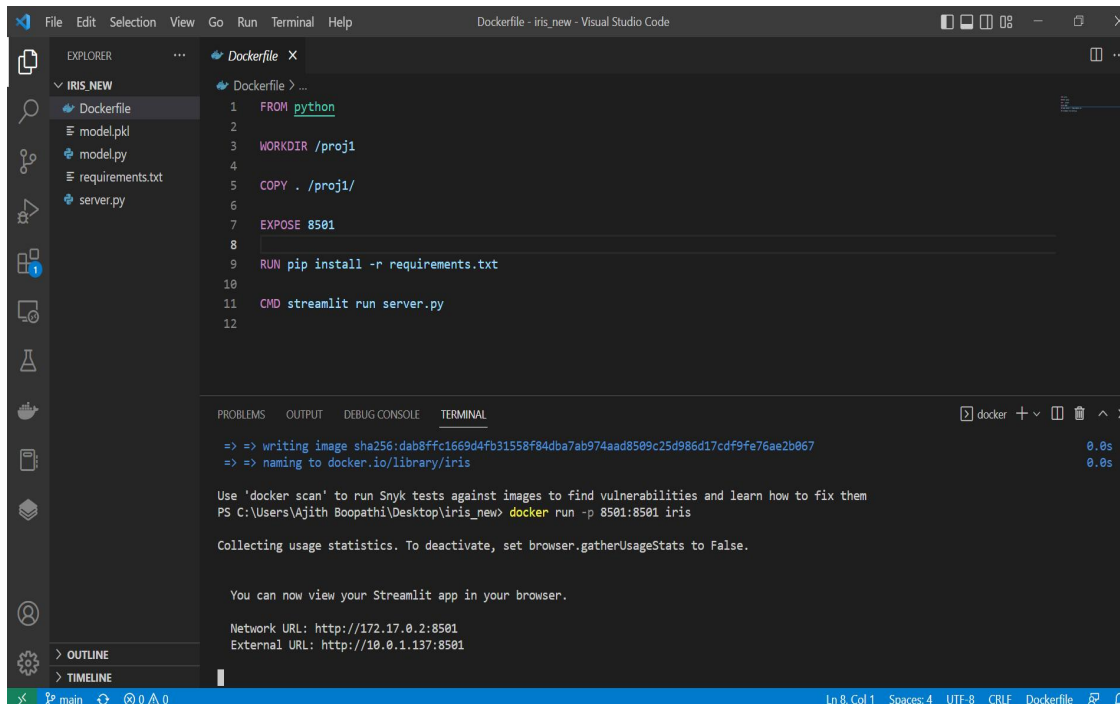
```

The terminal window shows the output of the command `docker build -t iris .`. The build process is successful, and the image is named `iris`. The terminal output includes the following lines:

```

PS C:\Users\Ajith Boopathi\Desktop\iris_new> docker build -t iris .
[+] Building 80.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 173B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:latest
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/4] FROM docker.io/library/python@sha256:a3c8c6766535f85f18e7304d3a0111de5288d73935bcf1b024217005ad5ce195
=> resolve docker.io/library/python@sha256:a3c8c6766535f85f18e7304d3a0111de5288d73935bcf1b024217005ad5ce195
=> sha256:db6ee1ace89767b16c65198b1cda89e2dfea79c64ee964d5eb6015a528f2b856 23.23MB / 23.23MB
=> sha256:a3c8c6766535f85f18e7304d3a0111de5288d73935bcf1b024217005ad5ce195 2.14kB / 2.14kB
=> sha256:cd972542db62c61e332c3a41eb78dc7d6bc74fc8bf98efbb6458e38ed73291a 2.22kB / 2.22kB
=> sha256:4c7b032ae8b7f24334a68d16f9698dca81fa03681426c110d47c330b39849de 3.06MB / 3.06MB
=> sha256:b44268c8cbc09471cdd86e2da2c3db6d41dd303b7a19560701945060bbda1d6a 8.58kB / 8.58kB

```



The screenshot shows the VS Code interface with the same Dockerfile open in the editor and a terminal window at the bottom. The terminal window shows the output of the command `docker run -p 8501:8501 iris`. The output includes the following lines:

```

=> => writing image sha256:dab8fffc1669d4fb31558f84dba7ab974aad8509c25d986d17cd9fe76ae2b067
=> => naming to docker.io/library/iris

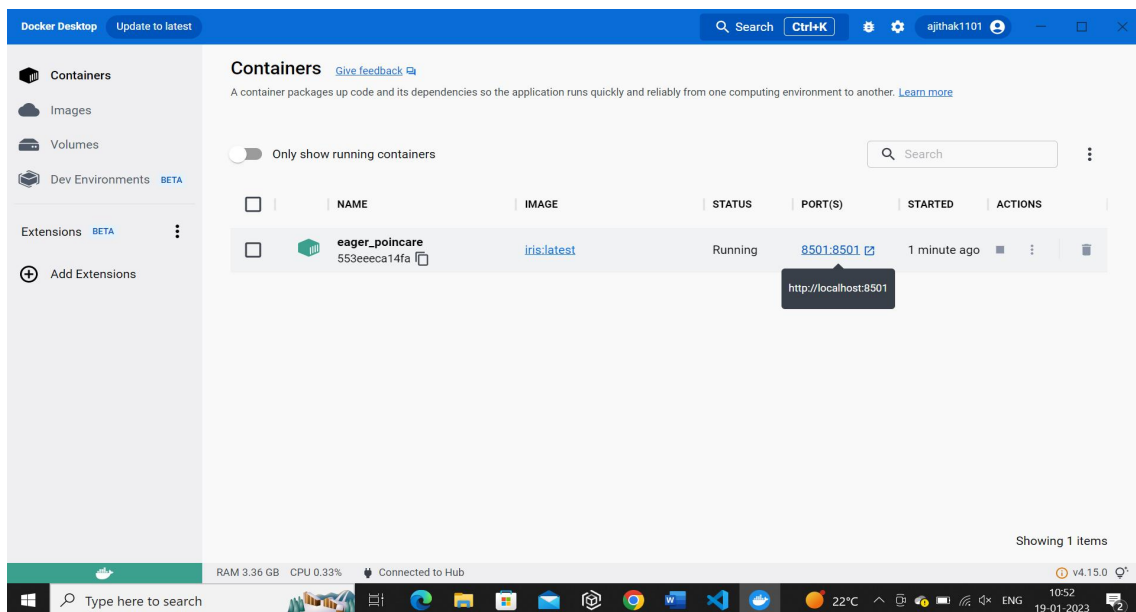
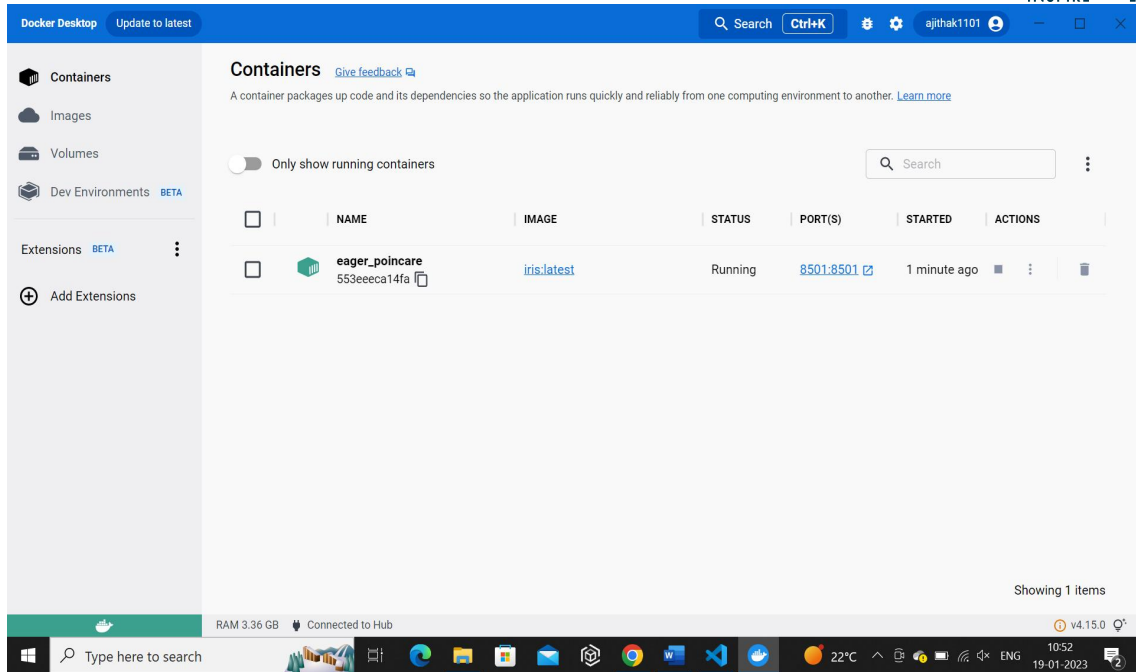
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\Ajith Boopathi\Desktop\iris_new> docker run -p 8501:8501 iris

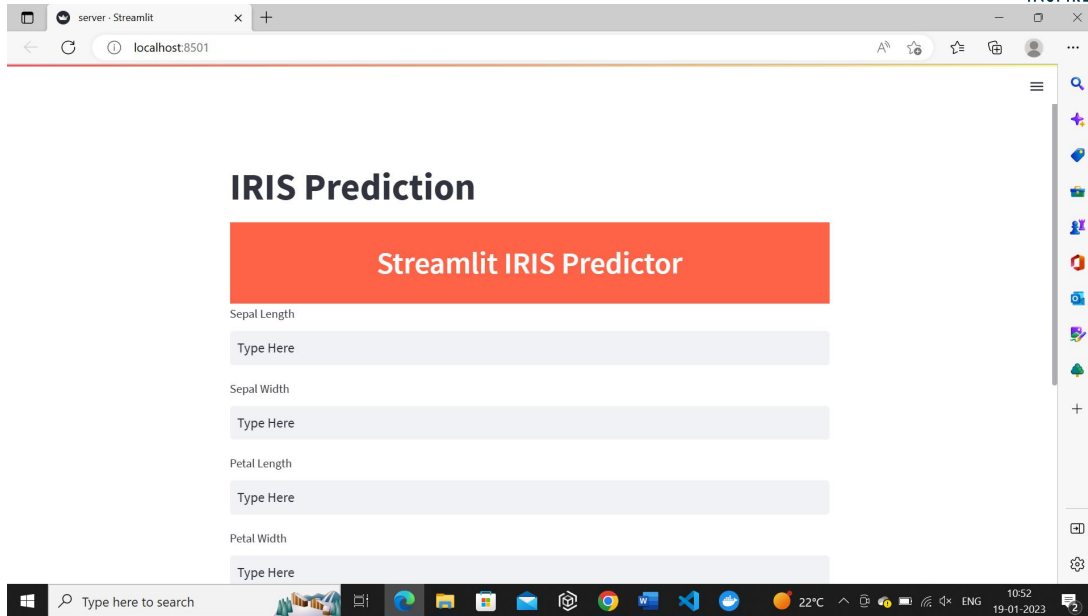
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.

You can now view your Streamlit app in your browser.

Network URL: http://172.17.0.2:8501
External URL: http://10.0.1.137:8501

```





IRIS Prediction

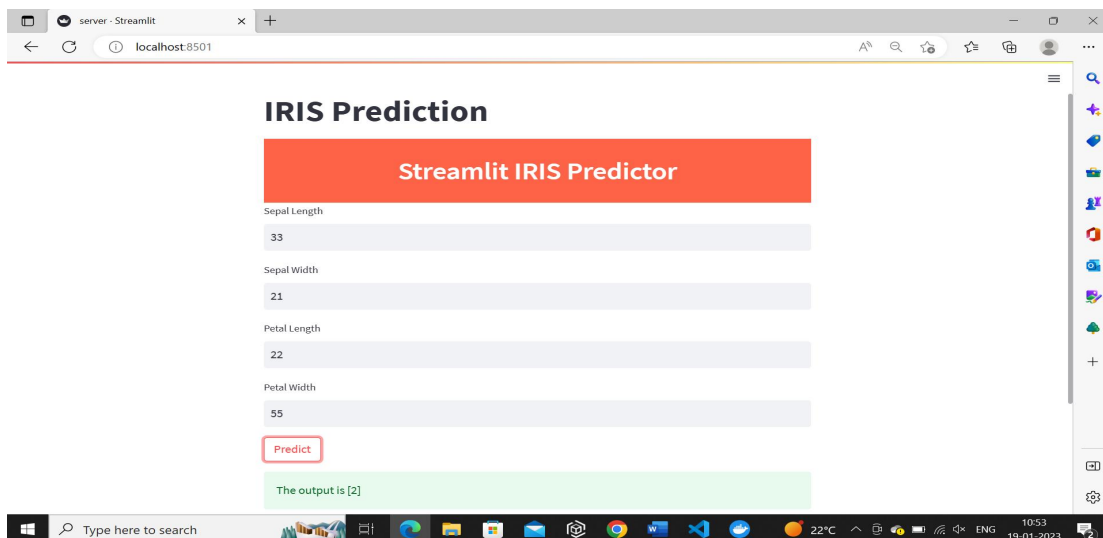
Streamlit IRIS Predictor

Sepal Length
Type Here

Sepal Width
Type Here

Petal Length
Type Here

Petal Width
Type Here



IRIS Prediction

Streamlit IRIS Predictor

Sepal Length
33

Sepal Width
21

Petal Length
22

Petal Width
55

Predict

The output is [2]

